The Examiner provides a tortured and unreasonable interpretation of the applied reference by presenting inconsistent positions in an attempt to reject the claims.

For example, the Examiner cites Microsoft Computer Dictionary for defining a linked list as "a list of nodes or ***elements of a data structure*** connected by pointers." Hence, the definition relied on by the Examiner requires a linked list to include: (1) a list of ***elements of a data structure***; (2) the elements of the data structure are ***connected***; and (3) the elements of the data structure are ***connected by pointers***. Consequently, a linked list does not include pointers in a data structure that refer to other data structures. See, for example, U.S. Patent No. 6,697,380 (of record in this application), which illustrates a first linked list of elements 56 of ***a data structure*** 27 that are linked by first pointers 62, and a second linked list of the elements 56 of the data structure 27 linked by second pointers 64.

As illustrated by the attached Exhibit A (page 1 of a definition of "Linked list" at http://en.wikipedia.org/wiki/Linked_list#Doubly-linked_lists by Wikipedia), linked lists also require linking to the same data type: "A linked list is a self-referential datatype because it contains a pointer or link to another data of the same type." The Microsoft and Wikipedia both are consistent in their requirement that the same data type is referenced, since storage of elements in the same data structure implies the elements are of the same type.

However, the Examiner disregards his own definition by ignoring the essential requirements of a linked list that the link (pointer) is to another entry in the same data structure and that links the same data type. In particular, the Examiner's analysis on pages 3-4 is fundamentally flawed because the Examiner simply asserts that "the process therefore forms a

linked list simply by referencing another entry by a pointer", while ignoring the essential teaching that the pointers should link to another element in the same data structure and be of the same type of element; rather, the pointers relied on by the Examiner are pointers between ***different data structure*** that have ***different data types***.

Avery stresses *repeatedly* that the DMA scoreboard is a special data structure that is ***distinct*** from the InfiniBand Address map 754. For example, the Abstract at lines 1-4 specifies that:

> Speculative prefetching during DMA reads in a message-passing, queue-oriented bus system is controlled by creating a special data structure, called a "DMA scoreboard", for each work queue entry associated with a DMA read.

(See also column 3, lines 18-20: "speculative prefetching is controlled by creating a special data structure, called a "DMA scoreboard", for each work queue entry associated with a DMA read with prefetching enabled.").

Further, Avery identifies the InfiniBand Address map 754 as a ***distinct data structure***:

> The InfiniBand address map 754 is a data structure that is stored locally in the InfiniBand-PCI bridge 324 and ***has a plurality of entries of which entries 756 and 758 are shown***. Each entry is associated with a region in the PCI address space 720 and holds the initial segment address for each region that is mapped into the system virtual memory address space, through the host channel adapter TPT.
>
> (Col. 8, lines 57-60).

Avery also suggests that the work queue pairs should be distinct data structures:

> HCA 308 [of Fig. 3] has a work queue pair consisting of send queue 310 and receive queue 312. Similarly, TCA 324 has a work queue pair consisting of send queue 326 and receive queue 328. Although only two queue pairs are shown, typically each client would create many more work queue pairs in order to conduct its operation. In order to use the work queue pair, a client submits a work request to its respective channel adapter and the work request causes an instruction called a Work Queue Entry (WQE) to be placed on the appropriate send work queue.

(Col. 6, lines 21-30).

Finally, Avery describes the DMA scoreboard 770 as a distinct data structure:

> The work queue entry 703 also contains a pointer 715 to a DMA context scoreboard 770. ***The DMA scoreboard 770 is a data structure*** that holds the DMA context and tracks outstanding DMA requests to insure that all outstanding requests are completed.
>
> (Col. 9, lines 58-62).

Based on the foregoing, there is no disclosure or suggestion whatsoever that the InfiniBand Address Map 754, the RDMA Work Queue Pair (consisting of work queue entries 703 and 705), and the DMA context scoreboard 770 would be implemented within the same data structure, especially since each of the Address Map 754, the work queue entries 703 and 705, and the DMA context scoreboard 770 have distinct data field types that have no similarly among each other.

Hence, the assertion that the address map 754, the work queue entries 703 and 705, andd the DMA context scoreboard 770 should be in the same data structure has no rational basis, and is inconsistent with the explicit teachings of Avery.

Hence, Avery fails to disclose or suggest generating any linked list whatsoever, in any form, because there is not a single link to another element of the same data structure. For this reason alone, the §102 rejection should be withdrawn because it fails to demonstrate that the applied reference discloses each and every element of the claim. See MPEP 2131. "The identical invention must be shown in as complete detail as is contained in the ... claim." Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). "Anticipation cannot be predicated on teachings in the reference which are vague or based on conjecture." Studiengesellschaft Kohle mbH v. Dart Industries, Inc., 549 F. Supp. 716, 216 USPQ 381 (D. Del. 1982), aff'd., 726 F.2d 724, 220 USPQ 841 (Fed. Cir. 1984).

In addition, there is no disclosure or suggestion whatsoever in Avery of: (1) a first linked list specifying a ***transmit sequence of transmitted work queue entries***; or (2) a second linked list

specifying an ***acknowledgement sequence of the transmitted work queue entries***, as claimed. In fact, there is no reference whatsoever to any linked list in Avery!

Anticipation cannot be established based on a piecemeal application of the reference, where the Examiner picks and chooses isolated features of the reference in an attempt to synthesize the claimed invention. "Anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim." *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). Hence, it is not sufficient that a single prior art reference discloses each element that is claimed, but the reference also must disclose that the elements are arranged as in the claims under review. *In re Bond*, 15 USPQ2d 1566, 1567 (Fed. Cir. 1990) (citing *Lindemann Maschinenfabrik GmbH*).

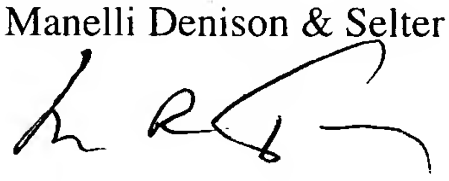
For these and other reasons, 102 rejection of claims 1 and 3-7 should be withdrawn.

It is believed dependent claim 2 is allowable in view of the foregoing.

In view of the above, it is believed this application is and condition for allowance, and such a Notice is respectfully solicited.

To the extent necessary, Applicant petitions for an extension of time under 37 C.F.R. 1.136. Please charge any shortage in fees due in connection with the filing of this paper, including any missing or insufficient fees under 37 C.F.R. 1.17(a) or 1.17(p), to Deposit Account No. 50-0687, under Order No. 95-507, and please credit any excess fees to such deposit account.

Respectfully submitted,
Manelli Denison & Selter PLLC



Leon R. Turkevich
Registration No. 34,035

Customer No. 20736
Date: December 27, 2005
(December 26, 2005 = Federal Holiday)

Linked list

From Wikipedia, the free encyclopedia.

In computer science, a **linked list** is one of the fundamental data structures used in computer programming. It consists of a sequence of nodes, each containing arbitrary data fields and one or two references ("links") pointing to the next and/or previous nodes. A linked list is a self-referential datatype because it contains a pointer or link to another data of the same type. Linked lists permit insertion and removal of nodes at any point in the list in constant time, but do not allow random access. Several different types of linked list exist: singly-linked lists, doubly-linked lists, and circularly-linked lists.

Linked lists can be implemented in most languages. Languages such as Lisp and Scheme have the data structure built in, along with operations to access the linked list. Procedural languages such as C, C++, and Java typically rely on mutable references to create linked lists.

Contents